I linguaggi di programmazione

Cos'è l'Informatica

Informatica = Informazione (dati + istruzioni) + Automazione.

Cos'è il Computer

Insieme organizzato di componenti, in grado di eseguire una successione di istruzioni finalizzate a elaborare l'informazione. È in grado di eseguire operazioni relativamente semplici ad alta velocità.

Cos'è un algoritmo

Sequenza **finita** e **ordinata** di passi/operazioni che portano alla realizzazione di uno specifico compito. L'algoritmo di un'operazione complessa può essere scomposto in una sequenza di istruzioni più semplici.

Esempi: calcoli matematici, massimo comune divisore, istruzioni di un elettrodomestico, prelevamento Bancomat, ricette di cucina.

N.B. Un computer è un esecutore di algoritmi

Proprietà di un algoritmo: **correttezza** (deve giungere alla soluzione del dato problema) ed **efficienza** (dare la soluzione nel modo più veloce, utilizzando la minima quantità di risorse fisiche).

Metodi di rappresentazione di un algoritmo:
- Linguaggio naturale
- Diagramma a blocchi
- Pseudo codice
- Linguaggio di programmazione

Cos'è un programma

Insieme di uno o più algoritmi scritti in un determinato linguaggio di programmazione. Il processore del computer esegue i programmi passo-passo in modo preciso e veloce.

Gli algoritmi sono parametrici

- Producono un risultato(*informazione*) che dipende da un insieme di dati di partenza

- Descrivono la soluzione non di un singolo problema, ma di una intera classe di problemi strutturalmente equivalenti

Esempi:

- l'algoritmo per la moltiplicazione di due numeri specifica come effettuare il prodotto di tutte le possibili coppie di numeri

- l'algoritmo per la ricerca di un libro nello schedario della biblioteca vale per tutti i possibili libri

Le istruzioni dell'algoritmo fanno riferimento a **variabili**, il cui valore non è fissato a priori ma cambia a seconda della situazione elaborativa in cui il processore del computer si trova.

Cos'è una variabile

E' un *dato*, formato dal tipo, un'etichetta e un valore, mantenuta all'interno della memoria RAM.

Il **tipo** di una variabile definisce come interpretare i dati memorizzati e lo spazio necessario alla sua memorizzazione (ad esempio numerico o stringa).

L'etichetta è il nome con cui riferirsi alla variabile all'interno del linguaggio di programmazione. In generale i linguaggi di programmazione distinguono minuscolo e maiuscolo (**case-sensitive**). L'etichetta deve rispettare le seguenti regole sintattiche: deve iniziare con una lettera minuscola o il

simbolo underscore (_) e può contenere lettere minuscole/maiuscole, cifre da 0 a 9 e il simbolo underscore (_).

Il **valore** che può assumere una variabile, in generale, dipende dal tipo della variabile.

Le variabili devono essere **dichiarate** prima di poter essere utilizzate, possono essere **inizializzate** all'atto della dichiarazione. All'atto della dichiarazione il tipo della variabile, se non è stato assegnato, è indefinito (**undefined**). L'inizializzazione assegna un valore ad una variabile e di conseguenza il tipo se non è stato definito precedentemente.

Javascript e PHP non permette di definire i tipi di variabili, sarà quindi il valore che andremo ad inserire a definire il tipo, per tale motivazione Javascript e PHP vengono detti linguaggi debolmente tipizzati (non fortemente tipizzati).

Uso delle variabili

- All'interno di espressioni (l'esecutore usa il valore contenuto nelle variabili per calcolare il risultato dell'espressione, per esempio op1 + op2 × op3 oppure op1 / op2 – op3, …)
- In istruzioni di assegnamento (introdurre nel contenitore identificato dal nome della variabile il valore specificato a destra dell'assegnamento, per esempio r = 35 (assegna 35 alla variabile il cui nome è r), pi = 3,14, …)
- In istruzioni di assegnamento combinate con espressioni (assegna a una variabile il risultato ottenuto dalla valutazione di un'espressione, per esempio in "circ = 2 × r × pi" il risultato dell'espressione 2 × r × pi viene calcolato utilizzando i valori contenuti nelle variabili r e pi e il risultato viene poi assegnato alla variabile circ; la stessa variabile può comparire in entrambi i lati dell'istruzione di assegnamento, per esempio in "k = k + 1" il valore contenuto in k viene utilizzato per trovare il valore dell'espressione k + 1 che viene memorizzato come nuovo valore di k.)

Assegnazione di valori a variabili

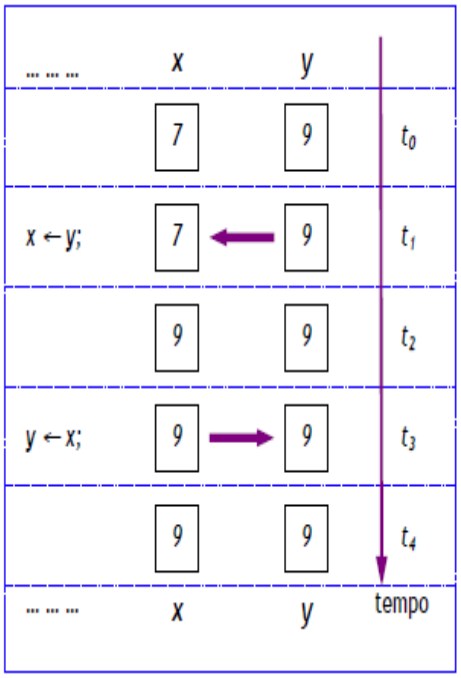
Il valore assegnato a una variabile si sostituisce a quello che era presente in precedenza, il vecchio valore non potrà più essere recuperato.

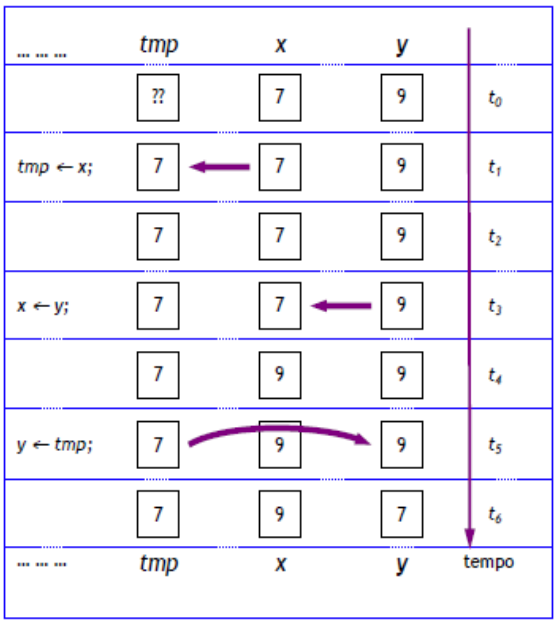
Esempio: si ipotizzi di voler scambiare i valori contenuti in due variabili x e y.

Soluzione proposta: doppio assegnamento del tipo

$$x \leftarrow y$$
$$y \leftarrow x$$

Per indicare che il valore di y deve essere copiato in x e che, nello stesso tempo, il valore di x sia trasferito in y. Le istruzioni però vengono eseguite in sequenza! Quindi l'assegnamento $x \leftarrow y$ viene completato prima di iniziare $y \leftarrow x$.





Soluzione corretta: uso di una variabile aggiuntiva (tmp), come strumento di memorizzazione temporanea ("buffer") del valore originariamente contenuto in x

$$tmp \leftarrow x$$
$$x \leftarrow y$$
$$y \leftarrow tmp$$

In questo modo lo scambio avviene senza perdere i valori originari.

Dati e Istruzioni

- Tipi di dati

– Numeri naturali o interi o reali (1, -2, 0.34)
– Caratteri alfanumerici (A, B, a, c, 1, 2, -, |, ...)
    - N.B. un insieme ordinato (l'ordine degli elementi ha importanza) di caratteri alfanumerici viene definita stringa (*il fatto che gli elementi dell'insieme abbiano un ordine non significa che gli elementi siano ordinati tra loro*).
– Dati logici o booleani (true, false)
– Array o vettore di n elementi ({1, 2, 3})
    - N.B. un array/vettore è un insieme ordinato (l'ordine degli elementi ha importanza) di elementi di un dato tipo (*il fatto che gli elementi dell'array abbiano un ordine non significa che gli elementi siano ordinati tra loro*). Essendo ordinato, è possibile accedere ad un elemento dell'array/vettore attraverso il suo indice, che corrisponde alla posizione all'interno dell'array/vettore.

• Istruzioni
– Operazioni di Input/Output (es. leggi, scrivi)
– Operazioni Aritmetico-logiche (es. somma = A + B)
– Strutture di controllo (es. SE, RIPETI)

• Struttura dati di tipo
– elementari (interi, alfanumerici, booleani, …)
– strutturati (array, matrici, …)

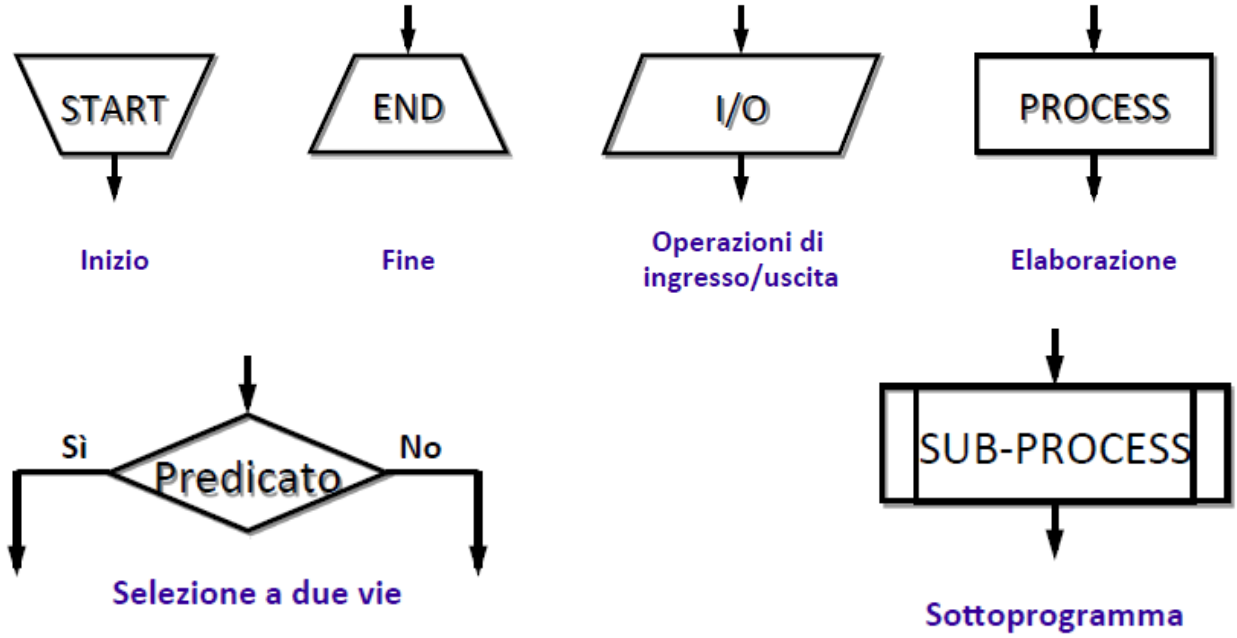
*N.B. Una **stringa** è un particolare array/vettore. Infatti è un insieme ordinato di singoli caratteri alfanumerici.*

Esempio di programma in linguaggio naturale: Determinare il maggiore tra due numeri

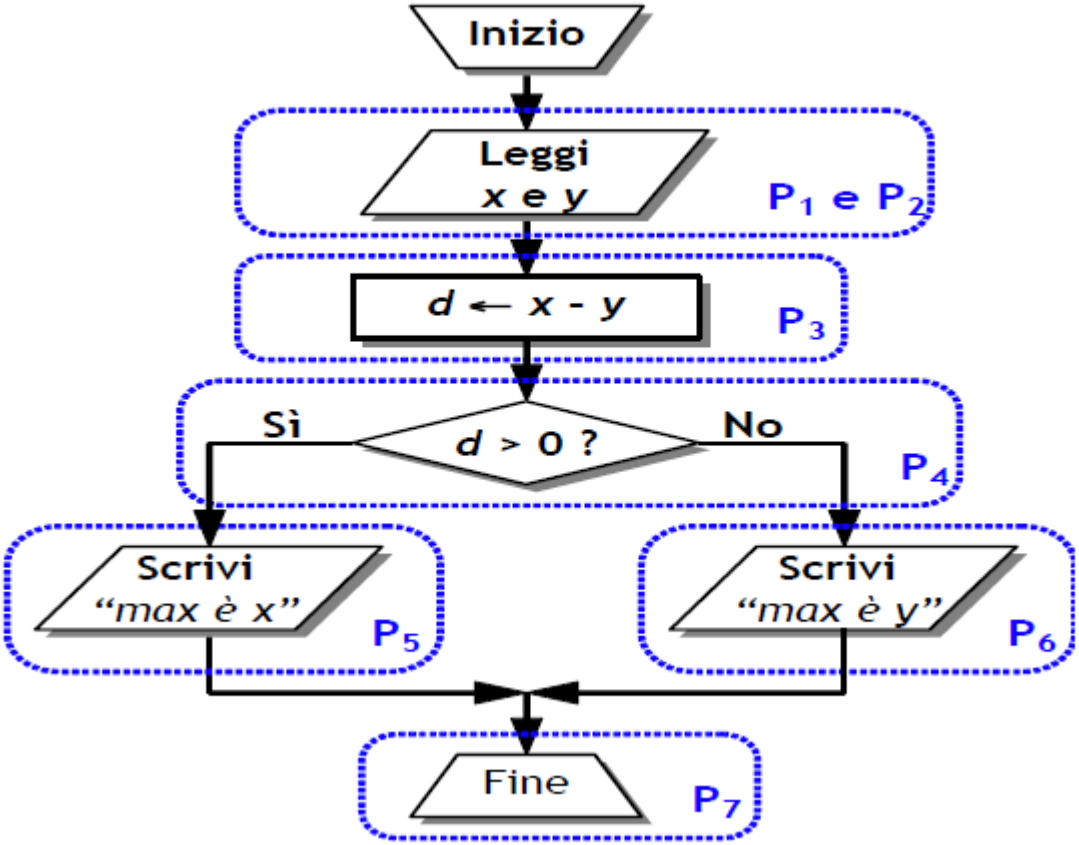
P1 leggi un valore dall'esterno e assegnalo alla variabile x;
P2 leggi un secondo valore dall'esterno e assegnalo alla variabile y;
P3 calcola la differenza d fra x e y, cioè esegui d ← x – y;
P4 valuta se d è positivo: in caso *affermativo* prosegui con il passo P5, altrimenti (in caso *negativo*) salta al passo P7;
P5 scrivi "il numero maggiore è " seguito dal valore di x;
P6 salta al passo P11;
P7 valuta se d è nullo: in caso *affermativo* prosegui con il passo P8, altrimenti (in caso *negativo*) salta al passo P10;

P8 scrivi “i due numeri sono uguali”;
P9 salta al passo P11;
P10 scrivi “il numero maggiore è ” seguito dal valore di y;
P11 termina l’esecuzione.

Struttura dei diagrammi di flusso



Diagrammi di flusso: ricerca massimo



Pseudo Codice: Prodotto di due numeri

```
Leggi alfa, beta;
prodotto ← 0;
Finché alfa > 0 ripeti
    prodotto ← prodotto + beta;
    alfa ← alfa – 1;
stampa prodotto;
```